Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №4 по курсу

«Операционные системы»

Группа: М8О-209БВ-24

Студент: Андреев М.В

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 16.12.25

Москва, 2025

Постановка задачи

Вариант 36.

Требуется создать динамические библиотеки, которые реализуют заданный вариантом функционал. Далее использовать данные библиотеки 2-мя способами: 1. Во время компиляции (на этапе линковки/linking) 2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками В конечном итоге, в лабораторной работе необходимо получить следующие части: Динамические библиотеки, реализующие контракты, которые заданы вариантом; Тестовая программа (программа №1), которая используют одну из библиотек, используя информацию полученные на этапе компиляции; Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты. Провести анализ двух типов использования библиотек. Пользовательский ввод для обоих программ должен быть организован следующим образом: Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»; “1 arg1 arg2 … argN”, где после “1” идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат ее выполнения; “2 arg1 arg2 … argM”, где после “2” идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат ее выполнения.

Функция 1:

8. Перевод числа x из десятичной системы счисления в другую:

Сигнатура функции: char *convert(int x);

Реализация №1: Перевод в двоичную

Реализация №2: Перевод в троичную

Функция 2:

9. Отсортировать целочисленный массив:

Сигнатура функции: int *sort(int *array, size_t n);

Реализация №1: Пузырьковая сортировка

Реализация №2: Сортировка Хоара

Общий метод и алгоритм решения

Использованные системные вызовы:

Системные вызовы для работы с динамическими библиотеками:

• dlopen(filename, flags) - загружает динамическую библиотеку в память процесса

• dlsym(handle, symbol) - получает адрес функции по её имени из библиотеки

• dlclose(handle) - выгружает библиотеку из памяти

• dlerror() - возвращает сообщение об ошибке работы с библиотеками

Функции компиляции и линковки:

• -fPIC - генерация позиционно-независимого кода для библиотек

• -shared - создание разделяемого объектного файла

• -L и -l - указание путей и имен библиотек при линковке

Алгоритм работы программы:

Программа №1 (статическая линковка):

1. Инициализация

- Компиляция программы с жесткой привязкой к библиотекам libtranslation_bin.so и libsort_bubble.so
- Программа содержит все необходимые библиотеки внутри исполняемого файла

2. Работа с пользовательским вводом

- Чтение команд пользователя через стандартный ввод
- Обработка команд:
- "1 <число>" - вызов функции translation() для перевода в двоичную систему
- "2 <массив чисел> 0" - вызов функции sort() для пузырьковой сортировки
- "q" - завершение работы программы

3. Выполнение операций

- Перевод числа: получение двоичного представления через фиксированную реализацию
- Сортировка массива: применение пузырькового алгоритма через фиксированную реализацию
- Вывод результатов пользователю

4. Завершение работы

- Освобождение выделенной памяти
- Возврат управления операционной системе

Программа №2 (динамическая загрузка):

1. Инициализация и загрузка библиотек

- Компиляция программы без привязки к конкретным библиотекам
- Загрузка четырех библиотек во время выполнения через dlopen():
- libtranslation_bin.so (двоичная система)
- libtranslation_ter.so (троичная система)
- libsort_bubble.so (пузырьковая сортировка)

- libsort_quick.so (сортировка Хоара)
- Получение указателей на функции через dlsym()

2. Работа с пользовательским вводом

- Чтение команд пользователя
- Обработка команд:
- "0" - переключение между реализациями функций
- "1 <число>" - вызов текущей реализации translation()
- "2 <массив чисел> 0" - вызов текущей реализации sort()
- "q" - завершение работы

3. Динамическое переключение реализаций

- При команде "0": смена текущих указателей на функции
- Выбор альтернативной реализации для обеих функций
- Обновление переменных current_translation и current_sort
- Информирование пользователя о новых реализациях

4. Выполнение операций через указатели

- Вызов функций через полученные указатели: current_translation(x) или current_sort(array)
- Обработка и вывод результатов, аналогично программе 1
- Освобождение памяти, выделенной библиотеками

5. Завершение работы

- Выгрузка всех библиотек через dlclose()
- Освобождение ресурсов программы

Сравнительный анализ

Статическая линковка (Программа 1):

- Библиотеки включаются в исполняемый файл на этапе компиляции
- Невозможно изменить реализацию без перекомпиляции
- Более простая архитектура, меньшие накладные расходы

Динамическая загрузка (Программа 2):

- Библиотеки загружаются во время выполнения программы
- Возможность "горячей" замены реализаций
- Гибкая архитектура, возможность обновления библиотек без перекомпиляции программы

Код программы

contract.h

```c
#ifndef CONTRACT_H
#define CONTRACT_H

#include <stddef.h>

#ifdef __cplusplus
extern "C" {
#endif

// Контракт для функции перевода числа
char* translation(long x);

// Контракт для функции сортировки
int* sort(int* array);

#ifdef __cplusplus
}
#endif

#endif
```

lib1_v1.c

```c
#include "../include/contract.h"
#include <stdlib.h>
#include <string.h>

#define BUF_SIZE 66

char* translation(long x) {
    if (x == 0) {
        char* res = (char *)malloc(2);
        if (!res) {
            return NULL;
        }
        strcpy(res, "0");
        return res;
    }

    char* buf = (char *)malloc(BUF_SIZE);
    if (!buf) {
        return NULL;
    }

    long temp = x;
    int is_negative = (x < 0);
    if (is_negative) {
        temp = -temp;
    }

    int i = BUF_SIZE - 2;
    buf[BUF_SIZE - 1] = '\0';

    while (temp > 0) {
```

```
        buf[i--] = (temp % 2) + '0';
        temp /= 2;
    }

    if (is_negative) {
        buf[i--] = '-';
    }

    char* res = strdup(&buf[i + 1]);
    free(buf);
    return res;
}
```

lib1_v2.c

```
#include "../include/contract.h"
#include <stdlib.h>
#include <string.h>

#define BUF_SIZE 66

char* translation(long x) {
    if (x == 0) {
        char* res = (char *)malloc(2);
        if (res == NULL) {
            return NULL;
        }
        strcpy(res, "0");
        return res;
    }

    long temp = x;
    int is_negative = (temp < 0);
    if (is_negative) {
        temp = -temp;
    }

    char* buf = (char *)malloc(BUF_SIZE);
    if (buf == NULL) {
        return NULL;
    }

    int i = BUF_SIZE - 2;
    buf[BUF_SIZE - 1] = '\0';

    while (temp > 0) {
        buf[i--] = (temp % 3) + '0';
        temp /= 3;
    }

    if (is_negative) {
        buf[i--] = '-';
    }
```

```
    char* res = strdup(&buf[i + 1]);
    free(buf);
    return res;
}
```

lib2_v1.c

```
#include "../include/contract.h"
#include <string.h>

// Находим размер массива (завершается 0)
static size_t array_size(int* array) {
    size_t size = 0;
    while (array[size] != 0) {
        size++;
    }
    return size;
}

int* sort(int* array) {
    size_t n = array_size(array);
    if (array == NULL || n == 0) {
        return array;
    }

    for (size_t i = 0; i < n - 1; i++) {
        for (size_t j = 0; j < n - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
    return array;
}
```

lib2_v2.c

```
#include "../include/contract.h"
#include <string.h>

// Находим размер массива (завершается 0)
static size_t array_size(int* array) {
    size_t size = 0;
    while (array[size] != 0) {
        size++;
    }
    return size;
}

static void swap(int *a, int *b) {
    int t = *a;
    *a = *b;
```

```
    *b = t;
}

static size_t partition(int *array, size_t low, size_t high) {
    int pivot = array[high];
    size_t i = low;

    for (size_t j = low; j < high; j++) {
        if (array[j] <= pivot) {
            swap(&array[i], &array[j]);
            i++;
        }
    }
    swap(&array[i], &array[high]);
    return i;
}

static void quick_sort_recursive(int *array, size_t low, size_t high) {
    if (low < high) {
        size_t pi = partition(array, low, high);

        if (pi > 0) quick_sort_recursive(array, low, pi - 1);
        quick_sort_recursive(array, pi + 1, high);
    }
}

int* sort(int* array) {
    size_t n = array_size(array);
    if (array == NULL || n < 2) return array;
    quick_sort_recursive(array, 0, n - 1);
    return array;
}
```

prog1_static.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include "../include/contract.h"

#define STDOUT_FD 1
#define STDIN_FD 0

static void print_str(const char* s) {
    write(STDOUT_FD, s, strlen(s));
}

static void print_int(int num) {
    char buf[32];
    int len = snprintf(buf, sizeof(buf), "%d", num);
    if (len > 0) {
        write(STDOUT_FD, buf, (size_t)len);
```

```
    }
}

static int parse_long(const char *str, long *val) {
    if (!str || !*str) return 0;
    long result = 0;
    int sign = 1;
    const char *p = str;

    while (*p && isspace(*p)) p++;

    if (*p == '-') {
        sign = -1;
        p++;
    } else if (*p == '+') {
        p++;
    }

    if (!isdigit(*p)) return 0;

    while (isdigit(*p)) {
        result = result * 10 + (*p - '0');
        p++;
    }

    *val = result * sign;
    return 1;
}

static void handle_function_1(const char* arg_str) {
    long x;
    if (!parse_long(arg_str, &x)) {
        print_str("Ошибка ввода. Для функции 1 требуется целое число\n");
        return;
    }

    char* res = translation(x);
    if (res) {
        print_str("Результат (двоичный): ");
        print_str(res);
        print_str("\n");
        free(res);
    } else {
        print_str("Ошибка выделения памяти\n");
    }
}

static void handle_function_2(const char* arg_str) {
    int temp_arr[101];  // +1 для завершающего нуля
    size_t count = 0;
    const char *p = arg_str;

    while (*p && count < 100) {
        long val;
```

```
        while (*p && isspace(*p)) p++;

        const char *start = p;
        if (*p == '-' || *p == '+') p++;
        while (*p && isdigit(*p)) p++;
        const char *end = p;

        if (end > start) {
            char num_str[30];
            size_t len = end - start;
            if (len >= sizeof(num_str)) len = sizeof(num_str) - 1;
            memcpy(num_str, start, len);
            num_str[len] = '\0';

            if (parse_long(num_str, &val)) {
                temp_arr[count++] = (int)val;
            }
        }

        if (*p == '\0' || end == start) break;
    }

    if (count == 0) {
        print_str("Ошибка ввода. Требуется список целых чисел\n");
        return;
    }

    temp_arr[count] = 0;  // Завершаем массив нулем

    int* arr_to_sort = (int *)malloc((count + 1) * sizeof(int));
    if (arr_to_sort == NULL) {
        print_str("Ошибка выделения памяти\n");
        return;
    }

    memcpy(arr_to_sort, temp_arr, (count + 1) * sizeof(int));

    print_str("Исходный массив: ");
    for (size_t i = 0; i < count; i++) {
        print_int(temp_arr[i]);
        print_str(" ");
    }
    print_str("\n");

    int* sorted_arr = sort(arr_to_sort);

    print_str("Результат (Пузырьковая): ");
    for(size_t i = 0; sorted_arr[i] != 0; i++) {
        print_int(sorted_arr[i]);
        print_str(" ");
    }
    print_str("\n");

    free(arr_to_sort);
```

```
}

int main() {
    char line[512];
    ssize_t bytes_read;

    print_str("Программа 1: Статическая линковка\n");
    print_str("Функция 1: Перевод в двоичную систему счисления\n");
    print_str("Функция 2: Пузырьковая сортировка\n");
    print_str("Введите команду: (1 [число] или 2 [массив чисел] или 'q' для выхода)\n");

    while ((bytes_read = read(STDIN_FD, line, sizeof(line) - 1)) > 0) {
        line[bytes_read] = '\0';

        if (line[bytes_read - 1] == '\n') {
            line[bytes_read - 1] = '\0';
        }

        if (line[0] == 'q') {
            break;
        }

        if (line[0] == '\0') continue;

        int cmd = line[0] - '0';
        char* args = line + 1;
        while (*args == ' ' || *args == '\t') {
            args++;
        }

        switch(cmd) {
            case 1:
                handle_function_1(args);
                break;
            case 2:
                handle_function_2(args);
                break;
            default:
                print_str("Недоступная команда, введите 1 или 2\n");
        }

        print_str("Введите команду: (1 [число] или 2 [массив чисел] или 'q' для
выхода)\n");
    }
    return 0;
}
```

prog2_dynamic.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <dlfcn.h>
```

```c
#include <unistd.h>
#include "../include/contract.h"

// Типы указателей на функции
typedef char* (*TranslationFunc)(long);
typedef int* (*SortFunc)(int*);

static TranslationFunc current_translation = NULL;
static SortFunc current_sort = NULL;

static void *lib_handle_1_v1 = NULL;
static void *lib_handle_1_v2 = NULL;
static void *lib_handle_2_v1 = NULL;
static void *lib_handle_2_v2 = NULL;

static int current_impl_1 = 1;  // 1 - двоичная, 2 - троичная
static int current_impl_2 = 1;  // 1 - пузырьковая, 2 - Хоара

#define LIB1_V1_PATH "bin/libtranslation_bin.so"
#define LIB1_V2_PATH "bin/libtranslation_ter.so"
#define LIB2_V1_PATH "bin/libsort_bubble.so"
#define LIB2_V2_PATH "bin/libsort_quick.so"

static void write_str(const char *str) {
    write(STDOUT_FILENO, str, strlen(str));
}

static void write_int(int num) {
    char buf[32];
    int len = snprintf(buf, sizeof(buf), "%d", num);
    if (len > 0) {
        write(STDOUT_FILENO, buf, (size_t)len);
    }
}

static int read_line(char *buffer, size_t max_len) {
    ssize_t bytes_read = read(STDIN_FILENO, buffer, max_len - 1);
    if (bytes_read <= 0) {
        return 0;
    }
    buffer[bytes_read] = '\0';

    char *newline = strchr(buffer, '\n');
    if (newline) {
        *newline = '\0';
    }

    return (int)bytes_read;
}

static int load_libraries() {
    lib_handle_1_v1 = dlopen(LIB1_V1_PATH, RTLD_LAZY);
    lib_handle_1_v2 = dlopen(LIB1_V2_PATH, RTLD_LAZY);
    lib_handle_2_v1 = dlopen(LIB2_V1_PATH, RTLD_LAZY);
```

```
    lib_handle_2_v2 = dlopen(LIB2_V2_PATH, RTLD_LAZY);

    if (!lib_handle_1_v1 || !lib_handle_1_v2 || !lib_handle_2_v1 || !lib_handle_2_v2) {
        char err_msg[512];
        snprintf(err_msg, sizeof(err_msg), "Ошибка загрузки одной из библиотек: %s\n",
dlerror());
        write(STDERR_FILENO, err_msg, strlen(err_msg));
        return 0;
    }

    current_translation = (TranslationFunc)dlsym(lib_handle_1_v1, "translation");
    current_sort = (SortFunc)dlsym(lib_handle_2_v1, "sort");

    if (!current_translation || !current_sort) {
        char err_msg[512];
        snprintf(err_msg, sizeof(err_msg), "Ошибка поиска символа 'translation' или
'sort': %s\n", dlerror());
        write(STDERR_FILENO, err_msg, strlen(err_msg));
        return 0;
    }

    return 1;
}

static void unload_libraries() {
    if (lib_handle_1_v1) dlclose(lib_handle_1_v1);
    if (lib_handle_1_v2) dlclose(lib_handle_1_v2);
    if (lib_handle_2_v1) dlclose(lib_handle_2_v1);
    if (lib_handle_2_v2) dlclose(lib_handle_2_v2);
}

static void switch_implementations() {
    // Переключаем реализацию перевода
    current_impl_1 = (current_impl_1 == 1) ? 2 : 1;
    void *target_handle_1 = (current_impl_1 == 1) ? lib_handle_1_v1 : lib_handle_1_v2;
    current_translation = (TranslationFunc)dlsym(target_handle_1, "translation");

    // Переключаем реализацию сортировки
    current_impl_2 = (current_impl_2 == 1) ? 2 : 1;
    void *target_handle_2 = (current_impl_2 == 1) ? lib_handle_2_v1 : lib_handle_2_v2;
    current_sort = (SortFunc)dlsym(target_handle_2, "sort");

    if (!current_translation || !current_sort) {
        write_str("Ошибка переключения: не удалось найти символ.\n");
        return;
    }

    write_str("--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---\n");

    char output[256];
    snprintf(output, sizeof(output), "Функция 1: %s\n", (current_impl_1 == 1) ? "Двоичная
(V1)" : "Троичная (V2)");
    write_str(output);
```

```
    snprintf(output, sizeof(output), "Функция 2: %s\n", (current_impl_2 == 1) ?
"Пузырьковая (V1)" : "Хоара (V2)");
    write_str(output);
}

static void handle_function_1(const char *arg_str) {
    if (!current_translation) {
        write_str("Ошибка: Функция 1 не загружена.\n");
        return;
    }

    long x;
    if (sscanf(arg_str, "%ld", &x) != 1) {
        write_str("Ошибка ввода: требуется целое число для функции 1.\n");
        return;
    }

    char *result = current_translation(x);

    if (result) {
        char output[256];
        snprintf(output, sizeof(output), "Результат (%s): %s\n",
            (current_impl_1 == 1) ? "Двоичный" : "Троичный", result);
        write_str(output);
        free(result);
    } else {
        write_str("Ошибка выделения памяти.\n");
    }
}

static void handle_function_2(const char *arg_str) {
    if (!current_sort) {
        write_str("Ошибка: Функция 2 не загружена.\n");
        return;
    }

    int temp_array[101];  // +1 для завершающего нуля
    size_t count = 0;
    const char *p = arg_str;

    while (sscanf(p, "%d", &temp_array[count]) == 1) {
        count++;
        while (*p && !isspace(*p)) p++;
        while (*p && isspace(*p)) p++;
        if (count >= 100) break;
    }

    if (count == 0) {
        write_str("Ошибка ввода: требуется список целых чисел для функции 2.\n");
        return;
    }

    temp_array[count] = 0;  // Завершаем массив нулем
```

```
    int *array_to_sort = (int*)malloc((count + 1) * sizeof(int));
    if (!array_to_sort) {
        write_str("Ошибка выделения памяти.\n");
        return;
    }

    memcpy(array_to_sort, temp_array, (count + 1) * sizeof(int));

    write_str("Исходный массив: ");
    for(size_t i = 0; i < count; i++) {
        write_int(temp_array[i]);
        write_str(" ");
    }
    write_str("\n");

    int *sorted_array = current_sort(array_to_sort);

    write_str("Результат (");
    write_str((current_impl_2 == 1) ? "Пузырьковая" : "Хоара");
    write_str("): ");

    for(size_t i = 0; sorted_array[i] != 0; i++) {
        write_int(sorted_array[i]);
        write_str(" ");
    }
    write_str("\n");

    free(array_to_sort);
}

int main() {
    if (!load_libraries()) {
        write_str("Критическая ошибка инициализации. Выход.\n");
        unload_libraries();
        return EXIT_FAILURE;
    }

    char line[512];

    write_str("--- Программа 2: Динамическая загрузка ---\n");
    write_str("Начальные реализации: Ф1 - Двоичная, Ф2 - Пузырьковая.\n");
    write_str("Введите команду (0 - переключить, 1 [число], 2 [массив чисел], q -
выход):\n");

    while (read_line(line, sizeof(line))) {
        if (line[0] == 'q' || line[0] == 'Q') break;

        int cmd = line[0] - '0';
        char *args = line + 1;
        while (*args == ' ' || *args == '\t') args++;

        switch (cmd) {
            case 0:
                switch_implementations();
```

```
                break;
            case 1:
                handle_function_1(args);
                break;
            case 2:
                handle_function_2(args);
                break;
            default:
                write_str("Неверная команда. Используйте 0, 1 или 2.\n");
                break;
        }

        write_str("\nВведите команду: ");
    }

    unload_libraries();
    return 0;
}
```

Протокол работы программы

Тестирование:

```
maxva@LAPTOP-RQH0OLUE:/mnt/d/Programming/OS/lab4$ ./bin/prog1_static
Программа 1: Статическая линковка
Функция 1: Перевод в двоичную систему счисления
Функция 2: Пузырьковая сортировка
Введите команду: (1 [число] или 2 [массив чисел] или 'q' для выхода)
1 56
Результат (двоичный): 111000
Введите команду: (1 [число] или 2 [массив чисел] или 'q' для выхода)
2 4 7 2 3 5
Исходный массив: 4 7 2 3 5
Результат (Пузырьковая): 2 3 4 5 7
Введите команду: (1 [число] или 2 [массив чисел] или 'q' для выхода)
q
maxva@LAPTOP-RQH0OLUE:/mnt/d/Programming/OS/lab4$ ./bin/prog2_dynamic
--- Программа 2: Динамическая загрузка ---
Начальные реализации: Ф1 - Двоичная, Ф2 - Пузырьковая.
Введите команду (0 - переключить, 1 [число], 2 [массив чисел], q - выход):
0
--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---
Функция 1: Троичная (V2)
Функция 2: Хоара (V2)

Введите команду: 1
Ошибка ввода: требуется целое число для функции 1.

Введите команду: 1 542
Результат (Троичный): 202002

Введите команду: q
```

Strace:

$ strace -f ./bin/prog2_dynamic

execve("./bin/prog2_dynamic", ["./bin/prog2_dynamic"], 0x7ffea3ddf8d8 /* 37 vars */) = 0

brk(NULL) = 0x555b28d2b000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f14de96a000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=37779, ...}) = 0

mmap(NULL, 37779, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f14de960000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f14de74e000

mmap(0x7f14de776000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f14de776000

mmap(0x7f14de8fe000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f14de8fe000

mmap(0x7f14de94d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f14de94d000

mmap(0x7f14de953000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f14de953000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f14de74b000

arch_prctl(ARCH_SET_FS, 0x7f14de74b740) = 0

set_tid_address(0x7f14de74ba10) = 16194

set_robust_list(0x7f14de74ba20, 24) = 0

rseq(0x7f14de74c060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f14de94d000, 16384, PROT_READ) = 0

mprotect(0x555af7a9f000, 4096, PROT_READ) = 0

mprotect(0x7f14de9a2000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f14de960000, 37779) = 0

getrandom("\x20\xfd\x32\xf8\xef\x19\xcb\x87", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x555b28d2b000

brk(0x555b28d4c000) = 0x555b28d4c000

openat(AT_FDCWD, "bin/libtranslation_bin.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15584, ...}) = 0

getcwd("/mnt/d/Programming/OS/lab4", 128) = 27

mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f14de965000

mmap(0x7f14de966000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f14de966000

mmap(0x7f14de967000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de967000

mmap(0x7f14de968000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de968000

close(3) = 0

mprotect(0x7f14de968000, 4096, PROT_READ) = 0

openat(AT_FDCWD, "bin/libtranslation_ter.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15584, ...}) = 0

getcwd("/mnt/d/Programming/OS/lab4", 128) = 27

mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f14de960000

mmap(0x7f14de961000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f14de961000

mmap(0x7f14de962000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de962000

mmap(0x7f14de963000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de963000

close(3) = 0

mprotect(0x7f14de963000, 4096, PROT_READ) = 0

openat(AT_FDCWD, "bin/libsort_bubble.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15144, ...}) = 0

getcwd("/mnt/d/Programming/OS/lab4", 128) = 27

mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f14de746000

mmap(0x7f14de747000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f14de747000

mmap(0x7f14de748000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de748000

mmap(0x7f14de749000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de749000

close(3) = 0

mprotect(0x7f14de749000, 4096, PROT_READ) = 0

openat(AT_FDCWD, "bin/libsort_quick.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15256, ...}) = 0

getcwd("/mnt/d/Programming/OS/lab4", 128) = 27

mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f14de741000

mmap(0x7f14de742000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f14de742000

mmap(0x7f14de743000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de743000

mmap(0x7f14de744000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f14de744000

close(3) = 0

mprotect(0x7f14de744000, 4096, PROT_READ) = 0

write(1, "--- \320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260 2: \320\224\320\270\320\275"..., 72--- Программа 2: Динамическая загрузка ---

) = 72

write(1, "\320\235\320\260\321\207\320\260\320\273\321\214\320\275\321\213\320\265 \321\200\320\265\320\260\320\273\320\270\320\267\320"..., 95Начальные реализации: Ф1 - Двоичная, Ф2 - Пузырьковая.

) = 95

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\272\320\276\320\274\320\260\320\275\320\264\321\203 (0"..., 121Введите команду (0 - переключить, 1 [число], 2 [массив чисел], q - выход):

) = 121

read(0, 0x7ffcb5d39f40, 511) = ? ERESTARTSYS (To be restarted if SA_RESTART is set)

--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---

read(0, 1 15

"1 15\n", 511) = 5

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 (\320\224\320\262\320\276\320\270\321\207\320\275"..., 44Результат (Двоичный): 1111

) = 44

write(1, "\n\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 32

Введите команду:) = 32

read(0, 0

"0\n", 511) = 2

write(1, "--- \320\240\320\225\320\220\320\233\320\230\320\227\320\220\320\246\320\230\320\230 \320\237\320\225\320\240\320"..., 52--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---

) = 52

write(1, "\320\244\321\203\320\275\320\272\321\206\320\270\321\217 1: \320\242\321\200\320\276\320\270\321\207\320\275\320\260"..., 40Функция 1: Троичная (V2)

) = 40

write(1, "\320\244\321\203\320\275\320\272\321\206\320\270\321\217 2: \320\245\320\276\320\260\321\200\320\260 (V2"..., 34Функция 2: Хоара (V2)

) = 34

write(1, "\n\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 32

Введите команду:) = 32

read(0, 1 54

"1 54\n", 511) = 5

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202
(\320\242\321\200\320\276\320\270\321\207\320\275"..., 44Результат (Троичный): 2000

) = 44

write(1, "\n\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 32

Введите команду:) = 32

read(0, q

"q\n", 511) = 2

munmap(0x7f14de965000, 16424) = 0

munmap(0x7f14de960000, 16424) = 0

munmap(0x7f14de746000, 16400) = 0

munmap(0x7f14de741000, 16400) = 0

exit_group(0) = ?

+++ exited with 0 +++

$ strace ./bin/prog1_static

execve("./bin/prog1_static", ["./bin/prog1_static"], 0x7ffd62165d60 /* 37 vars */) = 0

brk(NULL) = 0x55bdeb976000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f199b57f000

access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

**openat(AT_FDCWD, "./bin/glibc-hwcaps/x86-64-v3/libtranslation_bin.so", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (No such file or directory)**

**openat(AT_FDCWD, "./bin/glibc-hwcaps/x86-64-v2/libtranslation_bin.so", O_RDONLY|O_CLOEXEC)
= -1 ENOENT (No such file or directory)**

openat(AT_FDCWD, "./bin/libtranslation_bin.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15584, ...}) = 0

getcwd("/mnt/d/Programming/OS/lab4", 128) = 27

mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f199b57a000

**mmap(0x7f199b57b000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f199b57b000**

mmap(0x7f199b57c000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f199b57c000

mmap(0x7f199b57d000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f199b57d000

close(3) = 0

openat(AT_FDCWD, "./bin/glibc-hwcaps/x86-64-v3/libsort_bubble.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./bin/glibc-hwcaps/x86-64-v2/libsort_bubble.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./bin/libsort_bubble.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15144, ...}) = 0

getcwd("/mnt/d/Programming/OS/lab4", 128) = 27

mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f199b575000

mmap(0x7f199b576000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f199b576000

mmap(0x7f199b577000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f199b577000

mmap(0x7f199b578000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f199b578000

close(3) = 0

openat(AT_FDCWD, "./bin/glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./bin/glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./bin/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=37779, ...}) = 0

mmap(NULL, 37779, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f199b56b000

close(3) = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f199b359000

mmap(0x7f199b381000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f199b381000

mmap(0x7f199b509000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7f199b509000

mmap(0x7f199b558000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7f199b558000

mmap(0x7f199b55e000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f199b55e000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f199b356000

arch_prctl(ARCH_SET_FS, 0x7f199b356740) = 0

set_tid_address(0x7f199b356a10) = 17237

set_robust_list(0x7f199b356a20, 24) = 0

rseq(0x7f199b357060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f199b558000, 16384, PROT_READ) = 0

mprotect(0x7f199b578000, 4096, PROT_READ) = 0

mprotect(0x7f199b57d000, 4096, PROT_READ) = 0

mprotect(0x55bdb495b000, 4096, PROT_READ) = 0

mprotect(0x7f199b5b7000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f199b56b000, 37779) = 0

write(1, "\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260 1: \320\241\321\202\320\260\321\202\320\270"..., 62Программа 1: Статическая линковка

) = 62

write(1, "\320\244\321\203\320\275\320\272\321\206\320\270\321\217 1: \320\237\320\265\321\200\320\265\320\262\320\276\320\264"..., 87Функция 1: Перевод в двоичную систему счисления

) = 87

write(1, "\320\244\321\203\320\275\320\272\321\206\320\270\321\217 2: \320\237\321\203\320\267\321\213\321\200\321\214\320\272"..., 62Функция 2: Пузырьковая сортировка

) = 62

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \320\272\320\276\320\274\320\260\320\275\320\264\321\203: ("..., 114Введите команду: (1 [число] или 2 [массив чисел] или 'q' для выхода)

) = 114

read(0, 1 34

"1 34\n", 511) = 5

getrandom("\xf7\xac\x02\x3f\x73\x4b\xf2\x24", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x55bdeb976000

brk(0x55bdeb997000) = 0x55bdeb997000

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 (\320\264\320\262\320\276\320\270\321\207\320\275"..., 39Результат (двоичный):) = 39

write(1, "100010", 6100010) = 6

```
write(1, "\n", 1

)                   = 1

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ("..., 114Введите команду: (1 [число] или 2
[массив чисел] или 'q' для выхода)

) = 114

read(0, 2 3 2 8 5 4

"2 3 2 8 5 4\n", 511)          = 12

write(1, "\320\230\321\201\321\205\320\276\320\264\320\275\321\213\320\271
\320\274\320\260\321\201\321\201\320\270\320\262: ", 31Исходный массив: ) = 31

write(1, "3", 13)                    = 1

write(1, " ", 1 )                    = 1

write(1, "2", 12)                    = 1

write(1, " ", 1 )                    = 1

write(1, "8", 18)                    = 1

write(1, " ", 1 )                    = 1

write(1, "5", 15)                    = 1

write(1, " ", 1 )                    = 1

write(1, "4", 14)                    = 1

write(1, " ", 1 )                    = 1

write(1, "\n", 1

)                   = 1

write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202
(\320\237\321\203\320\267\321\213\321\200\321\214"..., 45Результат (Пузырьковая): ) = 45

write(1, "2", 12)                    = 1

write(1, " ", 1 )                    = 1

write(1, "3", 13)                    = 1

write(1, " ", 1 )                    = 1

write(1, "4", 14)                    = 1

write(1, " ", 1 )                    = 1

write(1, "5", 15)                    = 1

write(1, " ", 1 )                    = 1

write(1, "8", 18)                    = 1

write(1, " ", 1 )                    = 1

write(1, "\n", 1

)                   = 1

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ("..., 114Введите команду: (1 [число] или 2
[массив чисел] или 'q' для выхода)
```

```
) = 114
read(0, q
"q\n", 511)                = 2
exit_group(0)                      = ?
+++ exited with 0 +++
```

Вывод

В ходе работы успешно освоены принципы работы с динамическими библиотеками. Реализованы два подхода: статическая линковка (библиотеки включаются в исполняемый файл) и динамическая загрузка (библиотеки подключаются во время выполнения).

Практически применены системные вызовы: dlopen() для загрузки библиотек, dlsym() для получения указателей на функции, dlclose() для выгрузки. Реализован механизм переключения между алгоритмами (двоичная/троичная системы, пузырьковая сортировка/сортировка Хоара) без перекомпиляции программы.

Освоено создание модульной архитектуры с единым интерфейсом, позволяющей заменять реализации функций. Полученные навыки полезны для разработки расширяемых приложений с подключаемыми модулями.